APPLICATION

FOR

UNITED STATES LETTERS PATENT

APPLICANT NAME: Nitin Jhingan

TITLE: CONFIGURABLE FLAT FILE DATA MAPPING TO A DATABASE

DOCKET NO. END920030060US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

CONFIGURABLE FLAT FILE DATA MAPPING TO A DATABASE

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] This invention generally relates to data mapping, and more specifically, the invention relates to mapping data from an input file to a database.

Background Art

[0002] Computer applications are being used more and more. Today, particularly with the widespread availability of the Internet, businesses are able to make an ever increasing number of applications available worldwide twenty-four hours a day, seven days a week for use by customers, employees and suppliers. This presents important opportunities and challenges for businesses.

[0003] One challenge is to design the applications so that they can be used by different users who may access or input data to the applications using different data formats or files. Another challenge is to design the applications so that, in case users change the data formats used to input data to the applications, or it otherwise becomes advantageous to change the way data is input to the applications, the applications can be readily modified to accommodate new or different data formats and files.

SUMMARY OF THE INVENTION

[0004] An object of this invention is to improve methods and systems for mapping data from a source to a database.

[0005] Another object of the invention is to provide an extensible framework having a group of components, which can be readily modified or replaced, for handling various functions as data is mapped from a source to a database.

[0006] A further object of the present invention is to provide a framework, comprised of a group of independently adjustable components, to process data from one source, validate the

data, apply business rules on the data, load the data into a database, log the results of this whole procedure, and send mailings to configured parties.

[0007]   These and other objectives are attained with a method and framework for mapping data from a data source to a data destination. The method comprises the step of providing a plurality of components for performing defined functions to map the data from the source to the destination. These plurality of components perform the steps of (i) reading data from the source, (ii) processing the read data according to a set of rules, and (iii) loading the processed data into the destination. Preferably the plurality of components perform the further steps of (iv) verifying the integrity of the read data, and (v) logging results into a file. Each of the components operates independently of the other of the components.

[0008]   The preferred embodiment of the invention, described in detail below, provides a framework written in the java programming language to read a file and to verify the file data by, for example, checking for counts and/or data consistencies. The framework then processes the data, which may involve applying various business logic classes as applicable, and then loads the data into a database. The framework also logs the results into a file, which can then be emailed.

[0009]   Further benefits and advantages of the invention will become apparent from a consideration of the following detailed description, given with reference to the accompanying drawings, which specify and show preferred embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010]   Figure 1 schematically illustrates a processing framework in accordance with this invention.

[0011]   Figure 2 shows a computer network system with which the present invention may be used.

[0012]   Figure 3 is an example of a file that may be read by the framework of Figure 1.

END920030060US1   -2-

[0013]   Figure 4 shows an example of a formatter interface code that may be used by the framework.

[0014]   Figure 5 provides an example of an implementer interface code that may be implemented by the framework.

[0015]   Figure 6 shows the code for a logger feature that may be used by the framework.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0016]   With reference to Figure 1, the preferred embodiment of the invention provides a framework, generally referenced at 10, for mapping data from a source, represented at 12, to a database, represented at 14; and this framework comprises a plurality of components 16, 20, 22, 24 and 26 for performing various functions. Component 16 is provided to read a file, and component 20 is used to verify the file data by, for example, checking for counts and/or data consistencies. Component 22 is used to process the data, which may involve applying various business logic classes as applicable, component 24 is utilized for loading the data into a database 14, and component 26 is provided to log the results into a file, which can then be emailed.

[0017]   Component 16 may utilize any suitable procedure and is provided with suitable means to read data from a file. For example component 16 may be designed to read flat, or xml, files. Also, any suitable procedure may be employed by component 20 to verify the integrity of the read data. Numerous such procedures are known in the art and can be used in the practice of this invention.

[0018]   Component 22 may apply a wide range of business rules to the data read by framework. Preferably, these rules can be created, modified or deleted by an authorized administrator. In addition, any appropriate procedure may be used by component 24 for loading the data into a database. Many specific procedures are known by those of ordinary skill in the art and can be used in component 24.

END920030060US1   -3-

[0019]   It should be noted that while the specific framework 10 represented in Figure 1 only reads files, the framework can be extended to other sources of data like databases, mq series queues, etc. Also, while this specific embodiment of framework 10 only loads data into a database, the framework can be extended to other sources as well.

[0020]   Preferably, each of the components 16, 20, 22, 24 and 26 operates independently of the other components of the framework 10. In this way, each component can be modified or replaced without affecting the operation of the other components. Also, preferably the individual programs of the components 16, 20, 22, 24 and 26 can be updated or modified by authorized administrators during the use of framework 10.

[0021]   Framework 10 of this invention can be used in many specific situations. For example, the framework may be used in conjunction with, and be specifically designed to assist, some other application, which uses the data from database 16. In addition, framework 10 may be used on or with a computer network, such as a LAN, a WAN or the Internet.

[0022]   Figure 2 shows, for example, framework 10 used in conjunction with another application 30 available via the Internet 32 at a Web site 34. More particularly, Figure 2 shows a server 36 and a plurality of clients 40, 42, 44. Server 36 and clients 40, 42, 44 are connected to the Internet 32, and the server is provided with application 30, which can be accessed by clients 40, 42, 44 via the Internet. Server 36 is also provided with framework 10, which runs on the server to map input data from the clients to database 14, from where the data can be accessed by application 30.

[0023]   Figures 3-6 show, as more particular examples, specific file or code that can be used with or on framework 10. In particular, Figure 3 shows an example of a file rcvd for currencies that may be read be framework 10. As mentioned above, it should be noted that while the specific framework 10 represented in Figures 1 and 2 only reads files, the framework can be extended to other sources of data like databases, mq series queues, etc.

[0024]   More specifically, framework 10 maps fixed length fields in the input file to a java object type pofield. This framework reads the xml file. For a particular field, the framework determines the start position, the length, what database column the field maps to, which database tables the field maps to, whether the field can be updated or not, and what kind of formatter to be applied on this field. The framework calls the java class as defined within the <formatter> </formatter> tags. This java class implements the formatter interface, defined as;- package com.ibm.kestrel bridges. Figure 4 shows an example of a formatter interface code.

[0025]   The interface of Figure 4 defines one method: String format(String param). This method may be overridden by the implementations of this formatter class. This is useful, for example, for formatting data fields. For example, a source system may send a date in the format of yyyymmdd, but the database table may accept only a timestamp. By utilizing a formatter, the present invention eliminates the need for writing code for formatting a date within the business logic classes.

[0026]   Another file may be used to define an event handler. The event handler preferably implements the Implementer interface:- package com.ibm.kestral bridges. Figurer 5 shows an example of an Implementer interface code.

[0027]   The Implementer Interface of Figure 5 refers to a Hashtable. This Hashtable object contains a map of pofield objects and other classes, like the DBActions class that is used to manage database activity.

[0028]   The framework 10 also preferably manages transactions. A transaction is started when the first record is read, and the transaction is committed or rolled back when the record, as defined in a record parameter, is encountered. This helps to insulate programs with which framework 10 is used from managing a transaction.

[0029]   Preferably, framework 10 includes a Logger feature that can be used by implementing the Logger interface:- package com.ibm.kestrel bridges. Figure 6 shows, as an example, the code for a Logger feature.

END920030060US1   -5-

[0030] Using this logger feature, an application can add various notifications, and then finally send e-mail to a configured list of email ids. The Logger feature of figure 6 supports sending mail using "TO, CC and the BCC features of the mail.

[0031] While it is apparent that the invention herein disclosed is well calculated to fulfill the objects stated above, it will be appreciated that numerous modifications and embodiments may be devised by those skilled in the art, and it is intended that the appended claims cover all such modifications and embodiments as fall within the true spirit and scope of the present invention.